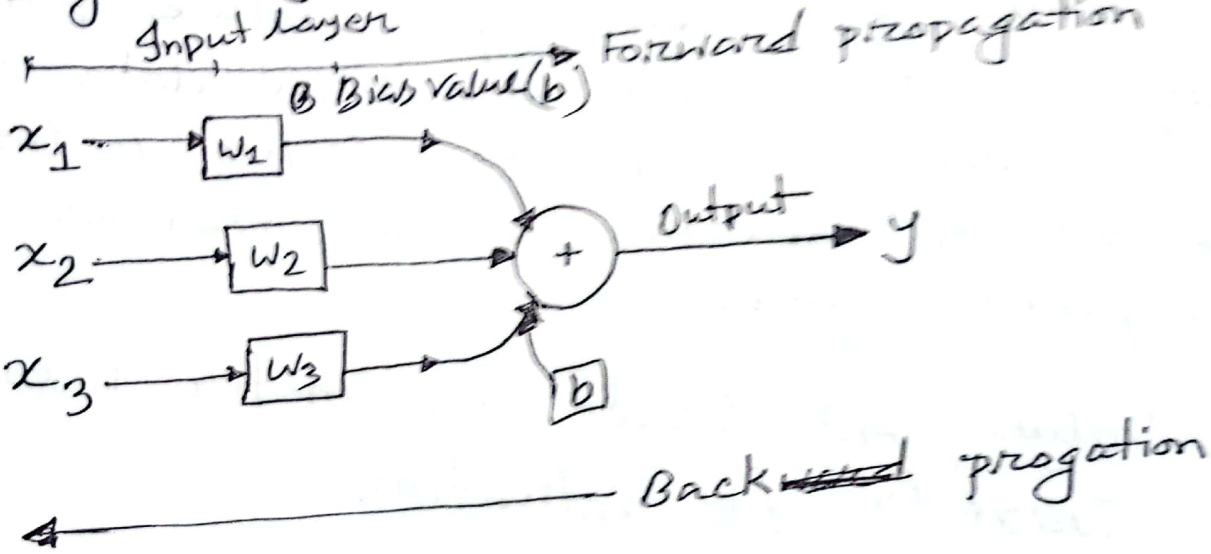
Artificial Neural Networks (ANNs):

→ Single Layer Perceptron (Artificial Neuron):



* An artificial neuron is a simple computational model that includes input layer with weights, an activation function and an output.

* Perceptron

Input layer: It is the initial layer where input data is enters the network. Each neuron in the input layer represents one feature or input variable.

Hidden layer:

Layers between the input and output layer responsible for learning complex patterns & representation from the input data. Here each neuron in a hidden layer is inter connected to neurons in the previous & subsequent layers

Bias Value: Additional parameter to control the threshold for activation.

* **Activation function:** Introduce non-linearity to the model. takes weighted sum of the inputs and produces an output by capturing complex patterns and relationships.

② **Loss function:** Difference between predicted value and actual value.

③ **Cost Function:** measures the overall error all training examples.

i) In regression... MSE.

ii) Binary classification... binary cross entropy

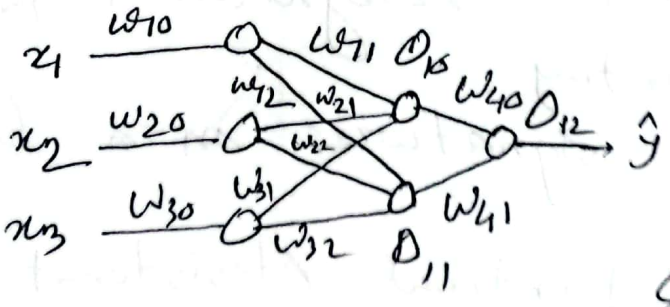
* Forward propation:

Process of passing input data through the networks layer, applying weights, biases, activation function etc to produce predictions.

(*) Back-ward propagation:

Process of calculating the gradient of loss with respect to the networks parameters & use this gradient to update the parameter using optimization algorithm.

* Chain Rule: used for weight update



* $W_{new} = W_{old} - \eta \frac{\delta L}{\delta W_{old}}$

where : $\eta \rightarrow$ learning rate

$\frac{\delta L}{\delta W_{old}} \rightarrow$ Slope

$$\frac{\delta L}{\delta w_{old}} = \frac{\partial L}{}$$

Weight update for: W_{44}

*① $W_{44new} = W_{44old} - \eta \frac{\delta L}{\delta W_{44old}}$

$$\frac{\delta L}{\delta W_{44old}} = \frac{\delta L}{\delta O_2} \times \frac{\delta O_2}{\delta O_{11}} \times \frac{\delta O_{11}}{\delta W_{11old}}$$

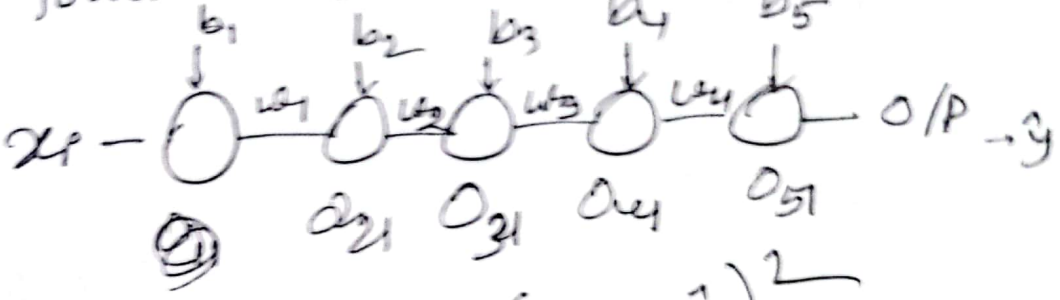
Output layer: final layer of the network that predicted output.

Real-world application of ANN:

i) Optical character recognization for data entry
ii) Validation of signatures on a bank cheque
iii) Siri, Alex (Virtual Assistant)

Vanishing Gradient Descent:

- If's becomes very small and tends towards zero.

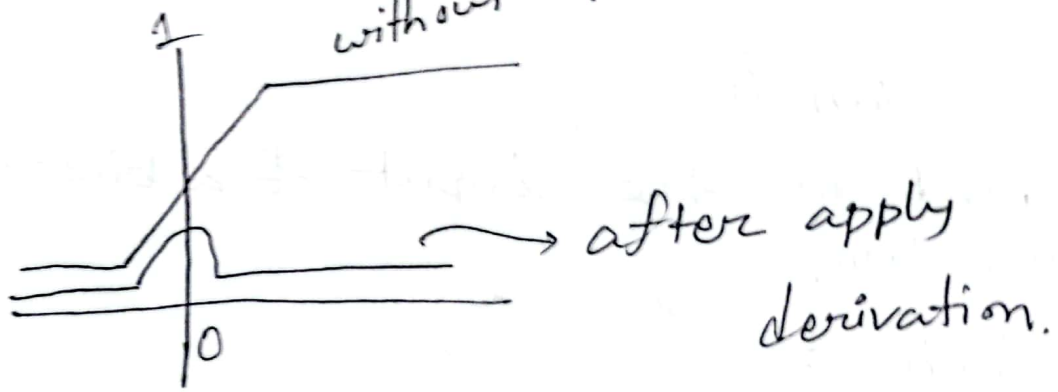


$$loss = \frac{1}{2}(y - \hat{y})^2$$

$$o_{51} = \sigma\left[(o_{41} \times w_4) + b_5\right]$$

↳ sigmoid

$w_{old} \approx w_{new}$



Kinds of Activation of function and their pros and Cons

① Sigmoid: Normally used as the output of a binary probabilistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

① pros

① Outputs are bounded between 0 and 1, which is useful for probabilistics.

② Cons:

* Not a zero centric function and computationally expensive.

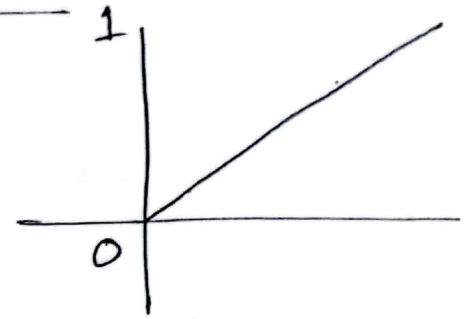
* Tan h

* used as the input of a binary probabilistic function;

* $\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

pros	cons
① Zero centric function	① Prone to vanishing Gradient function.

Relu: (Rectified linear Unit)



$$y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

pros	Cons
① Can deal with vanishing gradient problem.	→ Not a zero-centric function

② Softmax:

Used as the output in multi-class classification problems to find out different probabilities.

$$y = \frac{e^x}{\sum e^x}$$

pros	Cons
Outputs sum to 1	can saturate for inputs with large absolute values.

CNN

↳ CNN (Convalutional Neural Netwok).

Architecture:

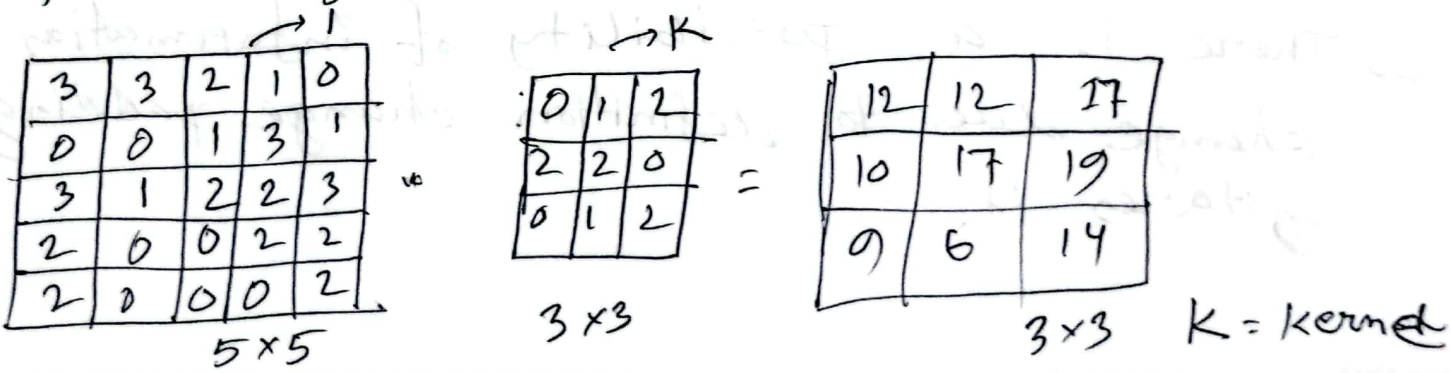
- kernel
- stride value
- padding
- pooling
- Flatten

Kernel:

→ Kernel is a filter, which is extracting feature from input data such as images.

→ Kernel is also a matrix, which is moving onto a input and its output also a matrix dot product.

→ Formula: output = $[i - k] + 1 = 3$

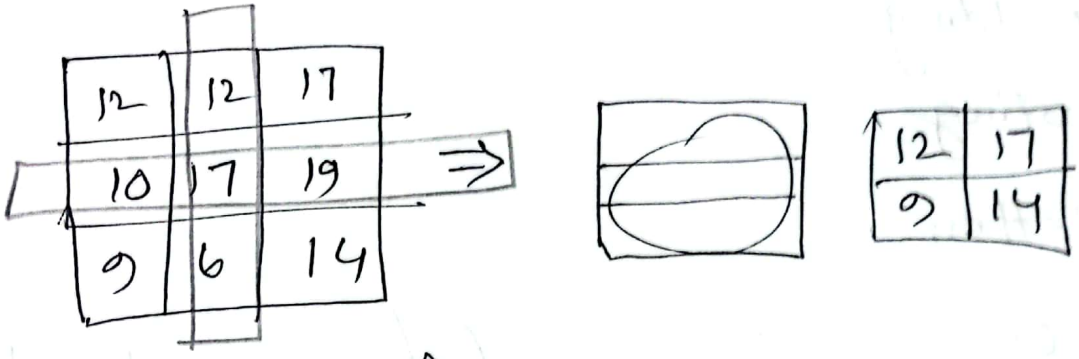


Stride Value:

→ Filter is moved accross the image left to right, top to bottom.

→ Amount of movement between application of the filter to the input image is referred to as the stride.

→



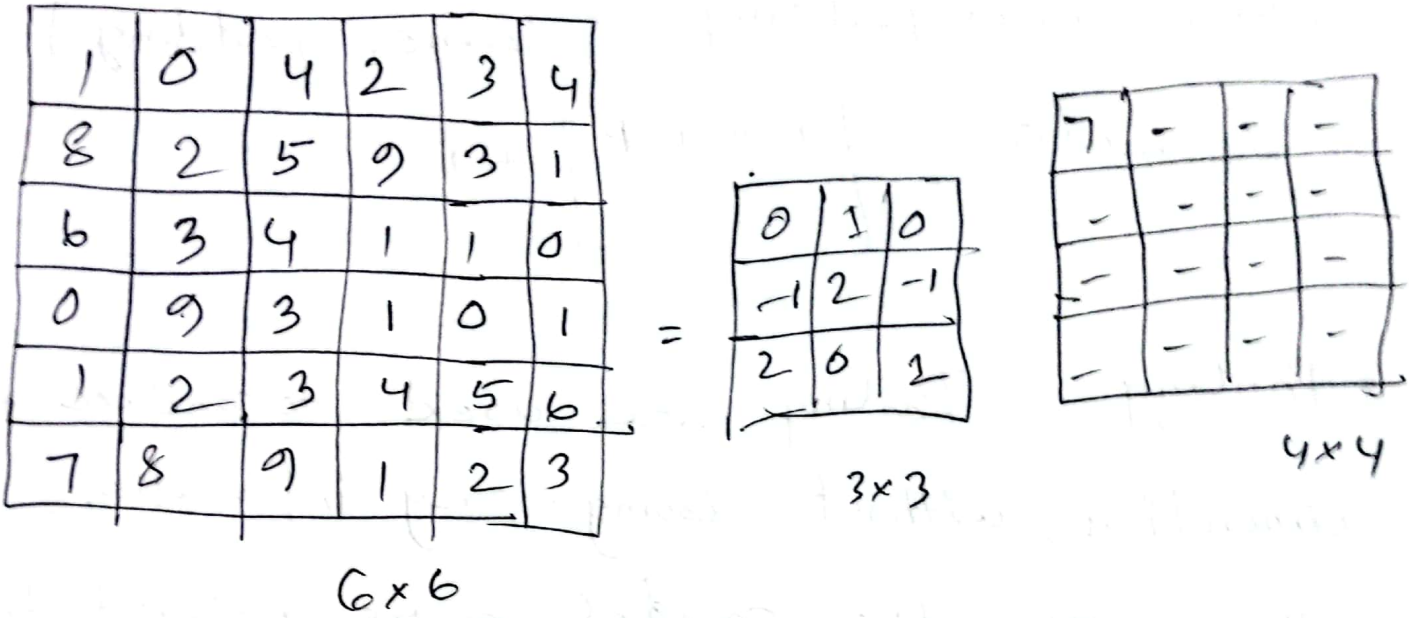
Ouput: $= \left[\frac{i-k}{s}\right] + 1$

$= \left[\frac{5-3}{2}\right] + 1 = 2$

Padding:

→ Padding fixes the border effect proble

→ There is a possibility of information change due to definition change padding stores it.



$1 \times 0 + 0 \times 1 + 4 \times 0 = 0$

$8 \times -1 + 2 \times 2 + 5 \times -1 = -9$

$6 \times 2 + 3 \times 0 + 4 \times 1 = 16$

$= 7$

without padding problem:

(i) Image shrinks

(ii) Corner pixels are less used compare the middle pixels.

adding zero padding (same padding)

formula = $\left[\frac{i-k+2p}{s}\right] + 1$

* Pooling: Pooling are used to reduce the dimention, without losing any information

→ It makes the model more robust to varians.

2	2	7	3
0	4	6	1
8	5	2	4
3	1	2	6

after padding

max Pool

0	7
8	6

Min Pool

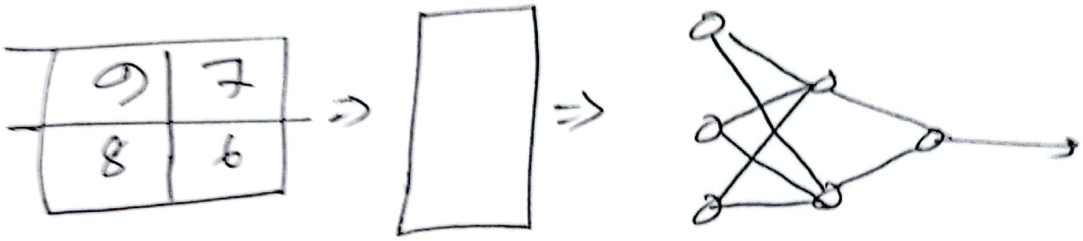
2	1
1	2

avg Pool

4.25	4.25
4.25	4.25

Flatten:

→ Transforming the entire pooled feature map / matrix into single column which is fed to the neural network for the processing.

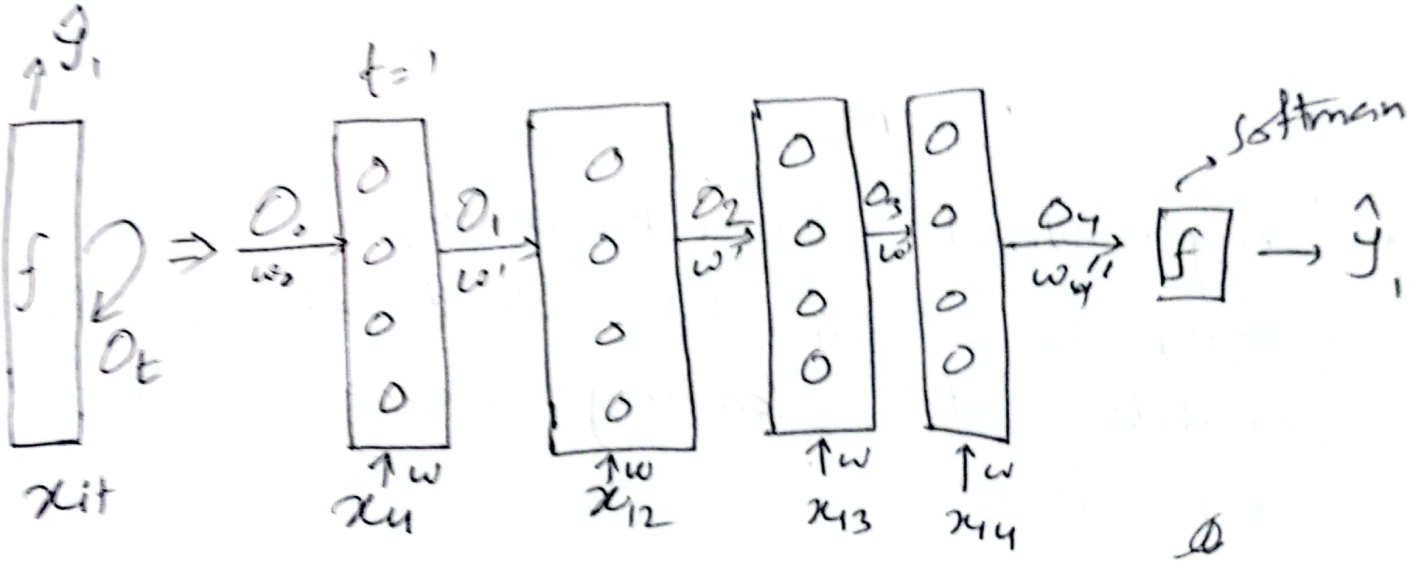


Application of CNN:

(i) Bio-metric identification of user identity
(ii) Unlocking smartphones and tablets.
(iii) Airport security and passport control.

RNN

- It is specialized neural networks for handling sequential data by maintaining a hidden state that captures the context and dependencies between elements in a sequence.
  - ↳ Application: NLP, speech recognition.



- $O_1 = f(x_{11} \cdot w)$
- $O_2 = f(x_{12} \cdot w + O_1 w_0')$
- $O_3 = f(x_{13} \cdot w + O_2 w')$
- $O_4 = f(x_{14} \cdot w + O_3 w')$

Weight Update:

$$W_{new} = W_{old} - \frac{\delta L}{\delta W_{old}}$$

$$\frac{\delta L}{\delta w''} = \left(\frac{\delta L}{\delta \hat{y}} * \frac{\delta \hat{y}}{\delta w''}\right)$$

$$\frac{\delta L}{\delta w'} = \left(\frac{\delta L}{\delta \hat{y}} * \frac{\delta \hat{y}}{\delta O_4} * \frac{\delta O_4}{\delta w'}\right)$$

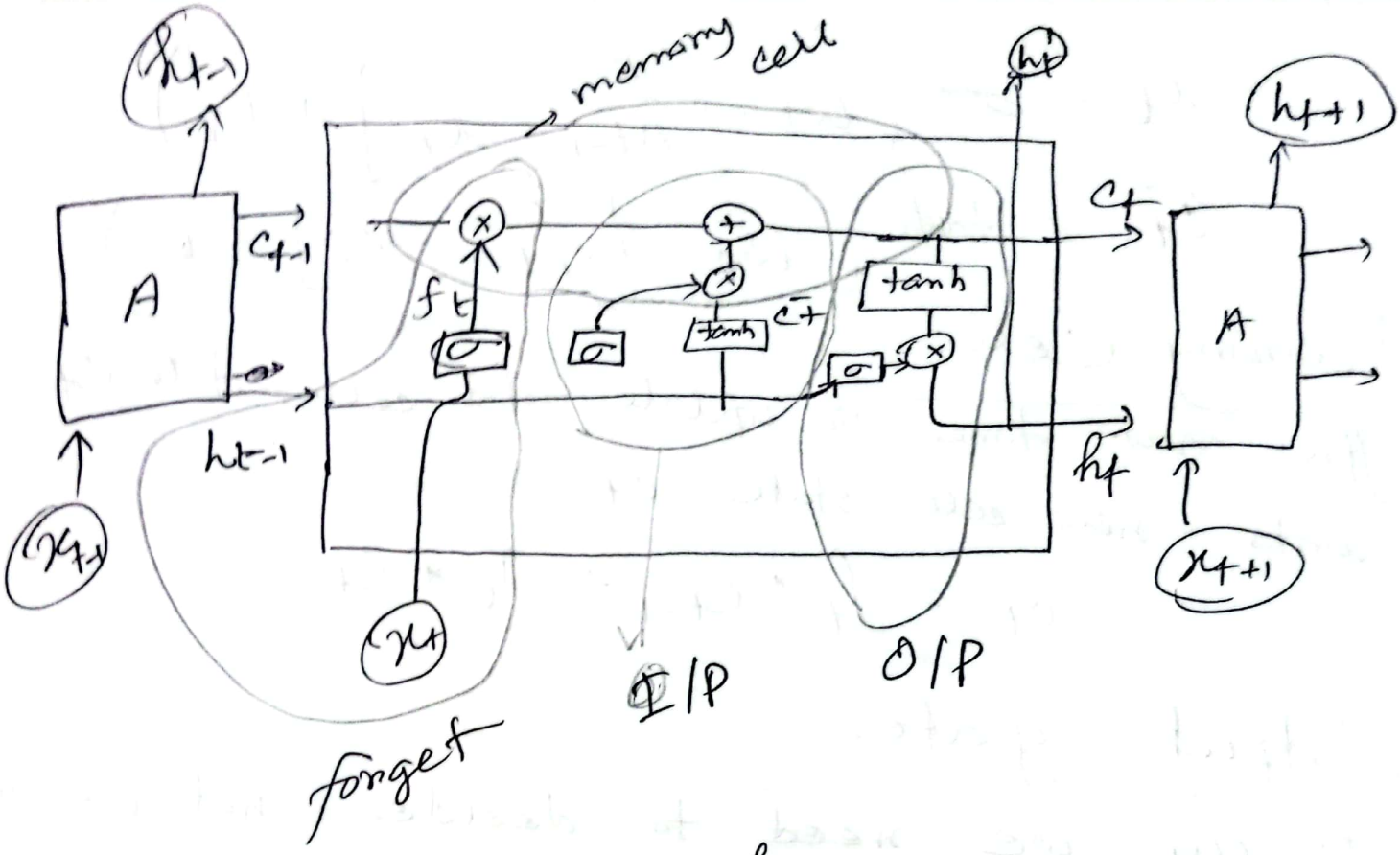
LSTM

→ Long - short term Memory

→ LSTM is a special kind of RNN capable of learning long-term dependencies.

* steps of LSTM:

(i) Memory cell
(ii) Forget gate
(iii) Input gate
(iv) Output gate.



① Forget gate / Sigmoid layer:

x_t and h_{t-1}

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

when we see a new subject, we want to forget the gender of the old subject.

Input gate:

We'd want to add the gender of the new subject to the cell state, to replace the old one we were forgetting.

$$i_t = \sigma\left(w_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\bar{c}_t = \tanh\left(w_c \cdot [h_{t-1}, x_t] + b_c\right)$$

Memory Cell:

Its now time to update old cell state c_{t-1} into new cell state c_t

$$c_t = f_t * c_{t-1} + i_t * \bar{c}_t$$

Output gate:

Finally, we need to decide what we are going to output, First sigmoid layer and put the cell state through the tanh and multiply it by the output of the sigmoid gate.

$$o_t = \sigma\left(w_o \cdot [h_{t-1}, x_t] + b_o\right)$$

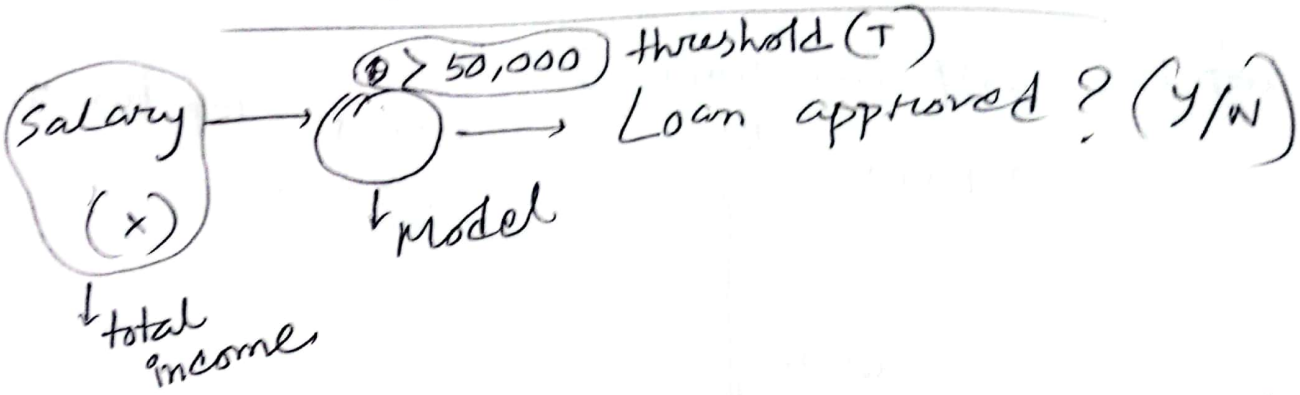
$$h_t = o_t * \tanh(c_t)$$

$\Rightarrow$

LSTM applications

(I) Speech recognition
(II) Rhythm learning
(III) Music composition
(IV) Grammer learning.

Understand Perceptron



Total income = Applicant Salary (x_1) + Father salary(x_2) + family member(x_3)

Total income = $x_1 + x_2 + x_3 > 0T$

$\Rightarrow x_1 + x_2 + x_3 - T > 0$

$\Rightarrow x_1 + x_2 + x_3 + bias > 0$

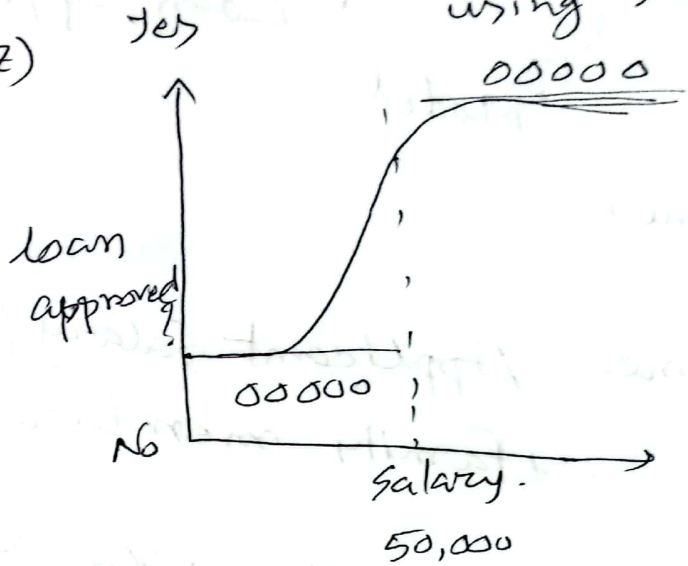
if $x_1 + x_2 + x_3 + b_i > 0 \rightarrow$ Output → 1

$x_1 + x_2 + x_3 + b_i \leq 0 \rightarrow$ Output → 0

Use of step-up function:

Output = step function (z)

= step (z)



* Perceptron Learning Algorithm:

→ If correct: Do nothing if the prediction input is equal to the target output.

→ If incorrect:

↳ scenario: (a) If output is 0 and target 1, add input vector to weight vector.

↳ scenario: (b) If output is 1 and target 0, subtract input vector from weight vector.

let,

$$D = \left(\left(x^{[1]}, y^{[1]} \right), \left(x^{[2]}, y^{[2]} \right), \ldots \left(x^{[n]}, y^{[n]} \right) \right) \in \left(\mathbb{R}^m \times \{0, 1\} \right)^n$$

A. for every $\left(x^{[i]}, y^{[i]} \right) \in D$:

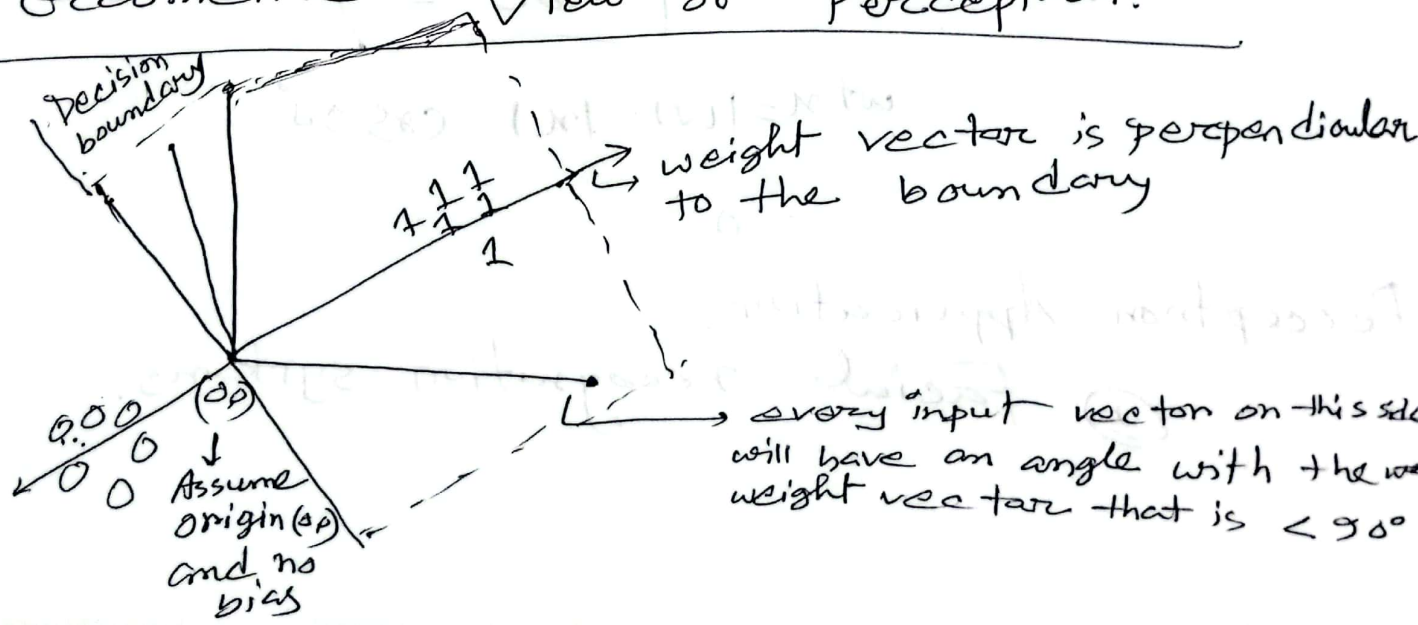
(a) $\hat{y}^{[i]} := \sigma\left(x^{[i]t} w \right)$

(b) $error := \left(y^{[i]} - \hat{y}^{[i]} \right)$

(c) $w_{new} := w_{old} + err \times x^{[i]}$

↳ weight update.

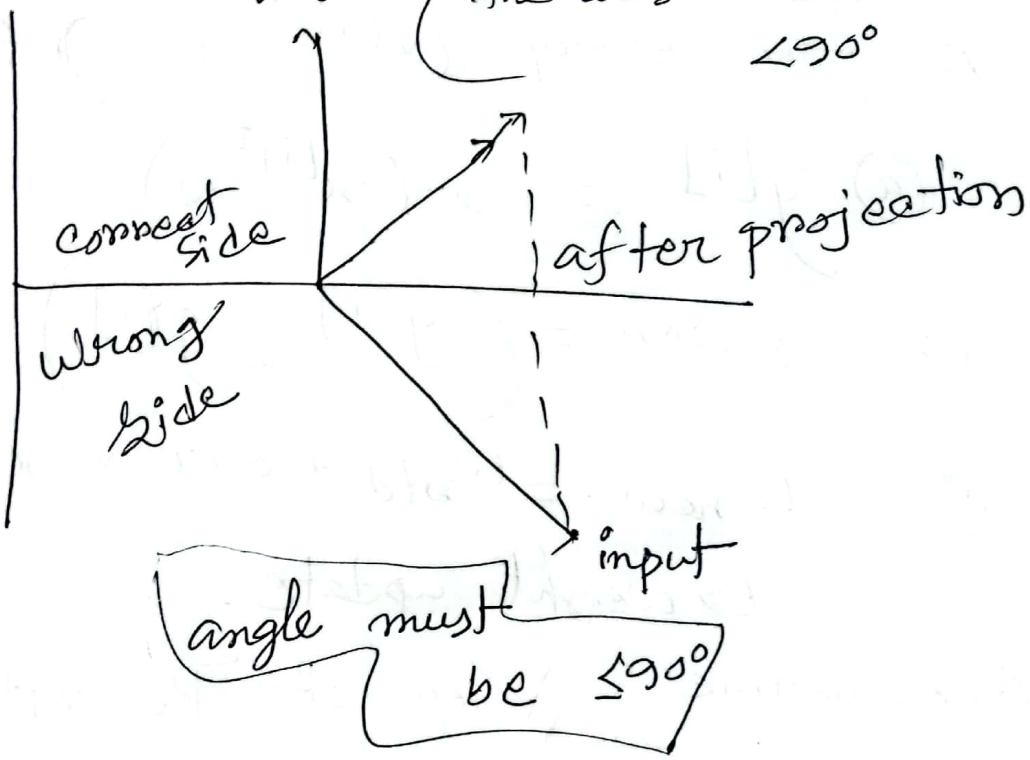
Geometric View of Perceptron:



$$\hat{y} = \begin{cases} 0 & w^t x \leq 0 \\ 1 & w^t x > 0 \end{cases}$$

$\cos 90^\circ = 1$

$w^t x = \|w\| \cdot \|x\| \cdot \cos\theta$ — so this needs to be 0 at the boundary it will have an angle with the weight vector that is $< 90^\circ$



$w^t x = |w|\ |x|\ \cos 90^\circ$

≥ 0

Perceptron Applications:

① Facial recognition systems.